Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 171 (2006) 693–707
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

An exact algorithm for the knapsack sharing problem with common items

Masako Fujimoto, Takeo Yamada *

Department of Computer Science, The National Defense Academy, 1-10-20 Hashirimizu, Yokosuka, Kanagawa 239-8686, Japan

Received 17 September 2003; accepted 30 September 2004
Available online 26 November 2004

Abstract

We are concerned with a variation of the knapsack problem as well as of the knapsack sharing problem, where we are given a set of n items and a knapsack of a fixed capacity. As usual, each item is associated with its profit and weight, and the problem is to determine the subset of items to be packed into the knapsack. However, in the problem there are s players and the items are divided into $s+1$ disjoint groups, N_k ($k = 0, 1, \ldots, s$). The player k is concerned only with the items in $N_0 \cup N_k$, where N_0 is the set of 'common' items, while N_k represents the set of his own items. The problem is to maximize the minimum of the profits of all the players. An algorithm is developed to solve this problem to optimality, and through a series of computational experiments, we evaluate the performance of the developed algorithm.

Keywords: Combinational optimization; Knapsack problem; Knapsack sharing

1. Introduction

In a previous work [17,18], we formulated the *knapsack sharing problem* (see also [6]) as an extension to the standard 0–1 *knapsack problem* [12,9] and proposed a solution algorithm to solve that problem. It was a combinatorial optimization problem [14] with a max-min type objective function, which has been widely studied in various frameworks [1,2,8,11,16,19,3]. In this paper we further extend the problem and formulate the knapsack sharing problem with common items, or the *generalized knapsack sharing problem* (GKSP), in the following way.

* Corresponding author. Tel.: +81 468 41 3810; fax: +81 468 44 5911.
E-mail address: yamada@nda.ac.jp (T. Yamada).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2004.09.036

As in the ordinary knapsack problem, we are given a set of n items $N := \{1, 2, \ldots, n\}$ and a knapsack of fixed capacity C. Associated with item $j \in N$ is its weight w_j and profit p_j. Furthermore, s 'players' are involved in this problem and items are divided into $s+1$ mutually disjoint subsets $N_0, N_1, \ldots, N_s$, i.e.,

$$\bigcup_{k=0}^{s} N_k = N, \quad N_k \cap N_l = \emptyset \ (k \neq l). \tag{1}$$

Here N_k is the set of player k's items ($k = 1, \ldots, s$). We assume that for each player all the items of other players are worthless. We call these *individual items*, and by $N_{\mathrm{I}} := \bigcup_{k=1}^{s} N_k$ we denote the set of all these items. Contrary, N_0 represents the set of *common items*; each of these is assumed to be of the identical worth to all the players.

Let $\boldsymbol{x} = (x_j) \in \{0,1\}^n$ be a solution vector, where $x_j = 1$ if item j is put into the knapsack, and $x_j = 0$ otherwise. Then, the value of $\boldsymbol{x}$ to player k is the sum of the profit from his own items and the profit from the common items, i.e.,

$$p^0(\boldsymbol{x}) + p^k(\boldsymbol{x}), \tag{2}$$

where

$$p^k(\boldsymbol{x}) := \sum_{j \in N_k} p_j x_j. \tag{3}$$

Our problem is to maximize the minimum of these profits over all the players subject to the usual weight and 0–1 constraints. Since $p^0(\boldsymbol{x})$ is common to all players, the problem can be written as follows.

GKSP:

$$\text{Maximize} \quad z(\boldsymbol{x}) := \min_{1 \leqslant k \leqslant s} \{p^k(\boldsymbol{x})\} + p^0(\boldsymbol{x}) \tag{4}$$

$$\text{subject to} \quad \sum_{j \in N} w_j x_j \leqslant C, \tag{5}$$

$$x_j \in \{0,1\}, \ \ j \in N. \tag{6}$$

Without much loss of generality, we assume the following:

A₁. Problem data C, p_j and $w_j (j \in N)$ are all positive integers.
A₂. $\sum_{j \in N} w_j > C$ and $w_j < C (j \in N)$.

GKSP is $\mathcal{NP}$-hard [5], since for $s = 0$ it reduces to the knapsack problem which is already $\mathcal{NP}$-hard [12]. Without common items (i.e., $N_0 = \emptyset$), GKSP is simply a knapsack sharing problem which is also $\mathcal{NP}$-hard [18].

In this paper we present a *decomposition* approach, where GKSP is solved to optimality by solving a knapsack problem (KP) and a knapsack sharing problem (KSP) parametrically. We implemented this algorithm, and evaluated this through a series of computational experiments.

2. Decomposition of the problem

First, we consider the following *auxiliary* knapsack problem.

$\mathrm{KP}_k(c)$:

$$\text{Maximize} \quad p^k(\boldsymbol{x}) \tag{7}$$

$$\text{subject to} \quad \sum_{j\in N_k} w_j x_j \leqslant c, \tag{8}$$

$$x_j \in \{0,1\}, \quad j \in N_k. \tag{9}$$

Let the optimal objective value of this problem be $z^{\star}_{\mathrm{KP}_k}(c)$. Specifically, for simplicity we write $\mathrm{KP}(c)$ and $z^{\star}_{\mathrm{KP}}(c)$ instead of $\mathrm{KP}_0(c)$ and $z^{\star}_{\mathrm{KP}_0}(c)$, respectively. Similarly, by $z^{\star}_{\mathrm{KSP}}(c)$ we denote the optimal objective value of the following knapsack sharing problem [18].

$\mathrm{KSP}(c)$:

$$\text{Maximize} \quad \min_{1\leqslant k\leqslant s}\{p^k(\boldsymbol{x})\} \tag{10}$$

$$\text{subject to} \quad \sum_{j\in N_{\mathrm{I}}} w_j x_j \leqslant c, \tag{11}$$

$$x_j \in \{0,1\}, \quad j \in N_{\mathrm{I}}. \tag{12}$$

Now, from the total knapsack capacity C let us allocate capacity c to the common items, and remaining $C-c$ to the individual items. Then, GKSP is solved by finding the value of c that maximizes the sum of $z^{\star}_{\mathrm{KP}}(c)$ and $z^{\star}_{\mathrm{KSP}}(C-c)$, i.e.,

$\mathrm{GKSP}^{\dagger}$:

$$\text{Maximize} \quad z^{\star}(c) := z^{\star}_{\mathrm{KP}}(c) + z^{\star}_{\mathrm{KSP}}(C-c) \tag{13}$$

$$\text{subject to} \quad 0 \leqslant c \leqslant C. \tag{14}$$

We note that both KP and KSP are $\mathcal{NP}$-hard, but for a fixed c these are relatively easy to solve in practice [12,18].

Let us consider $z^{\star}_{\mathrm{KP}_k}(c)$ and $z^{\star}_{\mathrm{KSP}}(c)$ as functions defined on $[0,\infty)$, and $c^{\star}_{\mathrm{KP}_k}(z)$ and $c^{\star}_{\mathrm{KSP}}(z)$ denote the 'inverse' functions of these respectively. More precisely, $c^{\star}_{\mathrm{KP}_k}(z)$ is defined as

$$c^{\star}_{\mathrm{KP}_k}(z) := \min\{c | z^{\star}_{\mathrm{KP}_k}(c) \geqslant z\} \tag{15}$$

and $c^{\star}_{\mathrm{KSP}}(z)$ is defined analogously. Then we have the following.

Theorem 1

(i) *$z^{\star}_{\mathrm{KP}_k}(c)$ and $z^{\star}_{\mathrm{KSP}}(c)$ are both monotonically non-decreasing, right-continuous step functions.*
(ii) *$z^{\star}_{\mathrm{KSP}}(c)$ is obtained by adding $z^{\star}_{\mathrm{KSP}_k}(c)$ $(1 \leqslant k \leqslant s)$ horizontally. That is,*

$$c^{\star}_{\mathrm{KSP}}(z) = \sum_{k=1}^{s} c^{\star}_{\mathrm{KP}_k}(z). \tag{16}$$

Proof. (i) For $z^{\star}_{\mathrm{KP}_k}(c)$, see [13]. The case of $z^{\star}_{\mathrm{KSP}}(c)$ is proved analogously. (ii) Let $c^{\star} := c^{\star}_{\mathrm{KSP}}(z)$ for $z \geqslant 0$. Then, by definition we have $z^{\star}_{\mathrm{KSP}}(c^{\star}) \geqslant z$. Let $\boldsymbol{x}^{\star}$ be an optimal solution to $\mathrm{KSP}(c^{\star})$, and define $c_k := \sum_{j\in N_k} w_j x^{\star}_j$. Then we have $z^{\star}_{\mathrm{KP}_k}(c_k) \geqslant z$ $(k=1,\ldots,s)$ and $\sum_{k=1}^{s} c_k \leqslant c^{\star}$. Note that $z^{\star}_{\mathrm{KP}_k}(c_k) \geqslant z$ is equivalent to $c^{\star}_{\mathrm{KP}_k}(z) \leqslant c_k$. Thus, we have

$$\sum_{k=1}^{s} c^{\star}_{\mathrm{KP}_k}(z) \leqslant c^{\star}_{\mathrm{KSP}}(z). \tag{17}$$

Next, let $c_k^\star := c_{\mathrm{KP}_k}^\star(z)$ for $z > 0$, and define $c^\star := \sum_{k=1}^s c_k^\star$. Then, we have $z_{\mathrm{KP}_k}^\star(c_k^\star) \geqslant z$, and altogether the set of solutions to $\mathrm{KP}_k(c_k^\star)$ $(k = 1, 2, \ldots, k)$ gives a feasible solution to $\mathrm{KSP}(c^\star)$ such that $z_{\mathrm{KSP}}^\star(c^\star) \geqslant z$. This implies

$$c_{\mathrm{KSP}}^\star(z) \leqslant \sum_{k=1}^{s} c_{\mathrm{KP}_k}^\star(z). \tag{18}$$

From (17) and (18), Theorem is proved. □

3. Lower and upper bounds

An upper bound to $\mathrm{KP}_k(c)$ is obtained by relaxing (9) to $0 \leqslant x_j \leqslant 1$. The optimal objective value, denoted as $\bar{z}_{\mathrm{KP}_k}(c)$, is easily obtained [10,12]. Similarly, by continuous relaxation of (12) we obtain an upper bound $\bar{z}_{\mathrm{KSP}}(c)$ to $\mathrm{KSP}(c)$ [18]. Furthermore, let $\underline{c}_{\mathrm{KP}_k}(z)$ and $\underline{c}_{\mathrm{KSP}}(z)$ denote the inverse functions of $\bar{z}_{\mathrm{KP}_k}(c)$ and $\bar{z}_{\mathrm{KSP}}(c)$ respectively. Then, we obtain the following [18].

Theorem 2

(i) $\bar{z}_{\mathrm{KP}_k}(c)$ *and* $\bar{z}_{\mathrm{KSP}}(c)$ *are both piecewise linear, monotonically non-decreasing, concave functions.*

(ii) $\bar{z}_{\mathrm{KSP}}(c)$ *is obtained by adding* $\bar{z}_{\mathrm{KP}_k}(c)(1 \leqslant k \leqslant s)$ *horizontally. That is,*

$$\underline{c}_{\mathrm{KSP}}(z) = \sum_{k=1}^{s} \underline{c}_{\mathrm{KP}_k}(z). \tag{19}$$

Let us define

$$\bar{z}(c) := \bar{z}_{\mathrm{KP}}(c) + \bar{z}_{\mathrm{KSP}}(C - c). \tag{20}$$

Then, since $\bar{z}_{\mathrm{KP}}(\cdot)$ and $\bar{z}_{\mathrm{KSP}}(\cdot)$ are both concave and piecewise linear, $\bar{z}(c)$ is also a concave and piecewise linear function. Let this function attain its maximum $\bar{z}$ at $\bar{c} \in [0, C]$. Then clearly $\bar{z}$ gives an upper bound to GKSP. Next, by solving $\mathrm{KP}(\bar{c})$ and $\mathrm{KSP}(C - \bar{c})$ exactly, we obtain a feasible solution to GKSP, and thus a lower bound

$$\underline{z} := z_{\mathrm{KP}}^\star(\bar{c}) + z_{\mathrm{KSP}}^\star(C - \bar{c}). \tag{21}$$

Example 1. Fig. 1 shows the functions $\bar{z}_{\mathrm{KP}}(c)$, $\bar{z}_{\mathrm{KP}_1}(C - c)$ and $\bar{z}_{\mathrm{KP}_2}(C - c)$, together with $\bar{z}_{\mathrm{KSP}}(C - c)$ and $\bar{z}(c)$ for a randomly generated instance with $n = 30$ and $s = 2$. The details of this example is available from our web site [20]. We note that $\bar{z}_{\mathrm{KSP}}$ is obtained as the horizontal sum of $\bar{z}_{\mathrm{KP}_1}$ and $\bar{z}_{\mathrm{KP}_2}$. Here we have an upper bound $\bar{z} = 8818$ at $\bar{c} = 1999$, and a lower bound $\underline{z} = 8666$.

4. An exact algorithm

Without loss of generality we assume $\underline{z} < \bar{z}$, since otherwise the problem is solved. Then, from concavity of $\bar{z}(c)$, the equation

$$\bar{z}(c) = \underline{z} \tag{22}$$

admits two distinct real solutions $\underline{c}_L$ and $\underline{c}_U$, where we assume $\underline{c}_L < \underline{c}_U$ (see Fig. 1). Then, since we already have a solution with the objective value $\underline{z}$, in solving $\mathrm{GKSP}^\dagger$ we only need to examine c within the interval

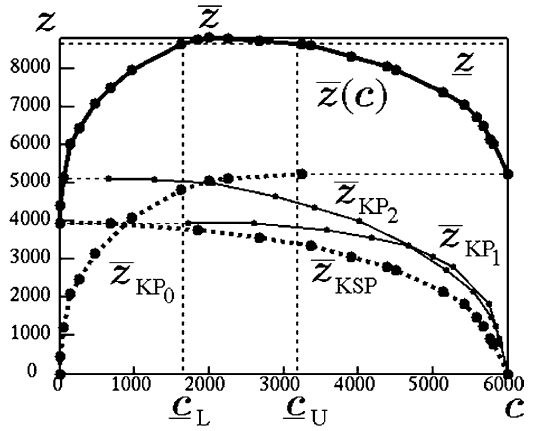

Fig. 1. Upper and lower bounds.

$[\underline{c}_L, \underline{c}_U]$. Moreover, since $z^{\star}_{\mathrm{KP}}(c)$ is a non-decreasing, right-continuous step function, it suffices to examine the *discontinuity points* of $z^{\star}_{\mathrm{KP}}(c)$. Discontinuity points can be found by the algorithm LISTUP_DC_POINTS given in Appendix A.

Thus, the algorithm to solve GKSP† is summarized as follows, where we start with the lower bound $\underline{z}$, which is taken as the initial *incumbent* solution.

Algorithm SOLVE_GKSP

At each discontinuity point $c \in [\underline{c}_L, \underline{c}_U]$ of $z^{\star}_{\mathrm{KP}}(\cdot)$, do the following:

Step 1. Solve KP(c) exactly to obtain $z^{\star}_{\mathrm{KP}}(c)$.
Step 2. Check, by the procedure to be stated below, if

$$z^{\star}_{\mathrm{KSP}}(C-c) \leqslant \underline{z} - z^{\star}_{\mathrm{KP}}(c) \tag{23}$$

holds. If YES go to **Step 4**.
Step 3. Update the lower bound $\underline{z}$ by

$$\underline{z} \leftarrow z^{\star}_{\mathrm{KP}}(c) + z^{\star}_{\mathrm{KSP}}(C-c). \tag{24}$$

Step 4. Go to the next discontinuity point c.

Example 2. For the case of Example 1, Fig. 2 depicts the functions $\bar{z}(c)$ and $z^{\star}_{\mathrm{KP}}(c)$ (see [20] for details). We have $\underline{c}_L = 1651.9$ and $\underline{c}_U = 3194.3$. Within the interval $[\underline{c}_L, \underline{c}_U]$, $z^{\star}_{\mathrm{KP}}(c)$ has four discontinuity points, and the problem is solved at $c^{\star} = 2261$ where $z^{\star}(c)$ attains the maximum $z^{\star} = 8719$.

Note that we do not necessarily need to solve KSP($C-c$) exactly to check, in Step 2, if (23) holds. To explain this, let us define

$$z^{\dagger} := \underline{z} - z^{\star}_{\mathrm{KP}}(c). \tag{25}$$

Then, (23) becomes

$$z^{\star}_{\mathrm{KSP}}(C-c) \leqslant z^{\dagger}, \tag{26}$$

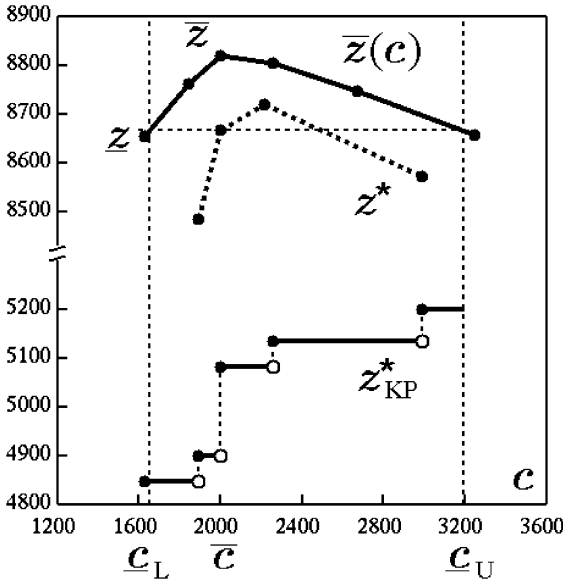

Fig. 2. The behavior of SOLVE GKSP(Generic).

which is equivalent to

$$c^{\star}_{\mathrm{KSP}}(z^{\dagger}) \geqslant C - c. \tag{27}$$

Instead of solving KSP($C - c$) exactly, we consider a less time consuming method to determine (26) or (27). First, we note that $c^{\star}_{\mathrm{KSP}}(z^{\dagger})$ can be obtained, by solving the inverse knapsack problems $\mathrm{IKP}_k(z^{\dagger})$ ($k = 1, 2, \ldots, s$) defined in Appendix A, as $c^{\star}_{\mathrm{KSP}}(z^{\dagger}) = \sum_{k=1}^{s} c^{\star}_{\mathrm{KP}_k}(z^{\dagger})$. Next, we have the following.

Theorem 3

$$c^{\star}_{\mathrm{KSP}}(z) \geqslant \sum_{k=1}^{s} \lceil \underline{c}_{\mathrm{KP}_k}(z) \rceil \geqslant \lceil \underline{c}_{\mathrm{KSP}}(z) \rceil. \tag{28}$$

Proof. Straightforward from Theorems 1 and 2. □

Based on this, we introduce the *check level* (CL) to determine (26) as follows.

- CL = 0: Determine (26) by solving KSP($C - c$) exactly.
- CL = 1: Determine (27) by solving $\mathrm{IKP}_k(z^{\dagger})$ exactly for $k = 1, \ldots, s$.
- CL = 2: Check if

$$\lceil \underline{c}_{\mathrm{KSP}}(z^{\dagger}) \rceil \geqslant C - c \tag{29}$$

  is satisfied first; if it fails then check (27) as in the case of CL = 1.
- CL = 3: Check (29) first; if it fails then check if

$$\sum_{k=1}^{s} \lceil \underline{c}_{\mathrm{KP}_k}(z^{\dagger}) \rceil \geqslant C - c \tag{30}$$

  holds, otherwise check (27).

For example, in the case of CL = 3 we first see if (29) holds. If it does, from (28) we have (27), and the answer to Step 2 of SOLVE_GKSP is known to be 'YES' without solving KSP exactly. Thus, by frequently bypassing the time consuming calculation to solve KSP exactly, the total time of computation is reduced.

5. Numerical experiments

We have implemented the solution algorithm of the previous section in C language and conducted some numerical experiments on an IBM RS/6000 Model 270 workstation. To solve KSP and KP inside the algorithm, we used the methods of Yamada et al. [18] and Horowitz and Sahni [7], respectively.

5.1. Design of experiments

Throughout this section weights and profits of items are assumed uncorrelated, and these are distributed uniformly over the interval $[1, 1000]$. We call this UNCOR type of problems. The number of items is between $n = 2^9$–2^{15}, the number of players is $s = 2$, 4 or 8, and the knapsack capacity is set to $C = 200n$. Since the average weight of items is 500.5, this means that about 40% of items can be accommodated into the knapsack. The ratio of the number of common items to n is denoted as $\lambda := |N_0|/n$, and we usually try $\lambda = 1/2, 1/4$ and $1/8$. The number of individual items is set to $|N_k| = (n - |N_0|)/s$ $(k = 1, \ldots, s)$.

5.2. Check level

Fig. 3 shows the CPU seconds vs. check level. Here $n = 2048$, $C = 200n$ and each measurement is the average over ten randomly generated instances. From the figure, the CPU time is shortest for CL = 3. Thus, from now on check level is fixed at CL = 3.

5.3. Comparison against an IP solver

We note that GKSP can be written as a linear 0–1 programming problem. Therefore, we compare our method against a commercial IP solver NUOPT [15], which is a product of a Japanese software vendor and is considered competitive to other popular solvers such as CPLEX, XPRESS-MP, or LINDO [4]. Fig. 4 shows the CPU time as a function of n for some values of λ and $s = 4$. Our method is much faster, and is able to solve larger problems than NUOPT.

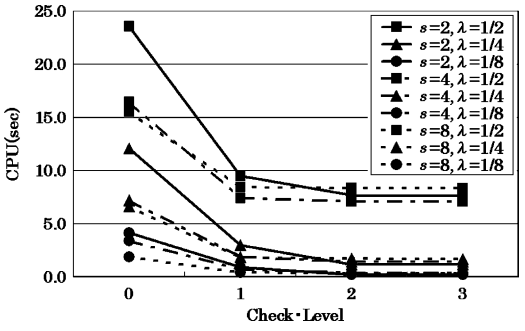

Fig. 3. Check level vs. CPU seconds ($n = 2048$).

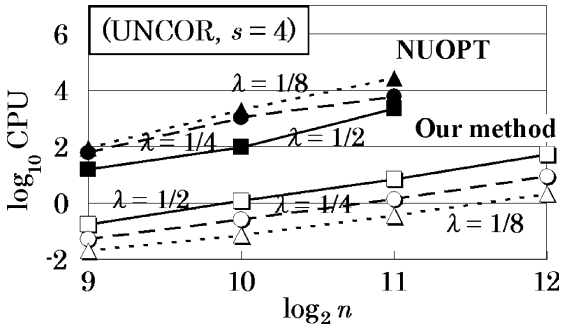

Fig. 4. Algorithm solve GKSP vs. NUOPT.

5.4. The result of experiments

Table 1 Panels (A)–(C) summarize the result of experiments for UNCOR instances. As in Figs. 3 and 4, each row is the average over ten randomly generated instances. Here $z^\star$ is the optimal objective value, ♯DC is the number of discontinuity points of $z^\star_{\mathrm{KP}}(\cdot)$ examined, and ♯KSP and ♯KP are, respectively, the numbers of KSP and KP solved to optimality. ♯KP includes the number of KPs solved within KSP as well.

From these tables we observe, approximately,

$$\text{CPU time} \approx n^{2.5}.$$

The CPU time decreases as λ decreases from 1/2 to 1/8, but it is rather insensitive to s. We also observe the following:

1. $z^\star$ increases linearly with n.
2. CPU time, $z^\star$, ♯KP and ♯DC all decrease as λ decreases from 1/2 to 1/8, while n and s are kept constant.
3. If we increase s from 2 to 8 while keeping n and λ constant, $z^\star$ decreases and ♯KP and CPU time increase, while ♯DC is relatively insensitive to s.

Fig. 5 depicts the CPU time as a function of λ when $n = 2048$ or 4096. We note that $\lambda = 0$ implies the knapsack sharing problem [18] since in this case we have no common items, and $\lambda = 1$ means the standard knapsack problem. GKSP is easily solved in practice in these extremal cases, while it is most difficult to solve for $\lambda \in [0.7, 0.9]$.

Finally, Fig. 6 shows the relation of the knapsack capacity to the CPU time in seconds. Here $n = 2048$, $\lambda = 1/2$ and the horizontal axis is α which is related to the knapsack capacity through $\alpha := C/n$. Since average weight of items is 500.5, $\alpha = 500$ means that almost all items can be accommodated into the knapsack in this case. The problem is most difficult when about a half of items can be included in the knapsack.

6. Numerical experiments: Correlated cases

In this section we introduce correlation between the weights and profits of items as follows [12].

- Weakly correlated case (WEAK)
  w_j: Uniformly and independently random over $[1, 1000]$.
  p_j: Uniformly and independently random over $[w_j, w_j + 200]$.

Table 1
The result of experiments

n	λ	$z^{\star}$	♯DC	♯KSP	♯KP	CPU (seconds)
Panel A: UNCOR, s = 2						
512	1/2	143,207.6	263.7	3.5	602.1	0.279
	1/4	118,747.4	76.2	1.6	137.6	0.039
	1/8	105,372.9	36.5	1.1	58.8	0.014
1024	1/2	287,017.7	504.4	3.6	1098.6	1.428
	1/4	238,262.9	179.0	1.6	287.0	0.229
	1/8	212,265.9	55.4	1.2	83.6	0.047
2048	1/2	580,009.8	895.8	3.3	1730.7	7.607
	1/4	478,850.5	348.4	3.0	496.7	1.151
	1/8	426,933.0	105.8	1.4	146.6	0.202
4096	1/2	1,160,495.6	1223.7	3.7	2441.8	38.745
	1/4	959,227.7	559.6	1.9	719.0	5.326
	1/8	852,130.8	259.0	1.8	330.6	1.222
8192	1/2	2,320,109.1	1896.9	4.0	3535.8	228.598
	1/4	1,917,876.8	771.8	1.6	1038.7	26.788
	1/8	1,704,748.9	389.6	1.5	462.5	4.111
16,384	1/2	4,635,270.9	2556.6	2.8	4307.3	1559.285
	1/4	3,835,638.7	1347.8	1.6	1808.5	183.417
	1/8	3,409,682.5	651.4	1.3	809.9	23.320
32,768	1/2	9,278,281.5	3526.9	3.1	5632.5	9027.214
	1/4	7,671,686.7	2042.0	2.0	2994.9	1532.052
	1/8	6,826,241.1	1025.7	1.6	1380.6	155.206
Panel B: UNCOR, s = 4						
512	1/2	129,161.2	194.5	4.2	926.8	0.180
	1/4	89,622.4	79.6	2.9	354.3	0.054
	1/8	67,943.7	32.5	1.7	143.4	0.021
1024	1/2	257,614.7	419.1	4.7	1926.8	1.202
	1/4	179,284.2	161.2	2.1	575.8	0.266
	1/8	136,604.4	58.9	1.1	185.9	0.073
2048	1/2	520,353.5	726.7	5.1	3259.6	7.078
	1/4	359,129.1	268.8	2.8	963.4	1.395
	1/8	274,233.0	109.6	1.6	318.1	0.363
4096	1/2	1,042,916.8	1382.1	5.0	6287.3	53.839
	1/4	720,578.1	560.3	2.6	1782.2	8.908
	1/8	546,848.4	244.7	1.6	646.0	2.144
8192	1/2	2,085,952.3	1949.2	5.3	8538.4	329.111
	1/4	1,442,061.1	914.1	2.9	2710.5	51.042
	1/8	1,095,595.0	386.2	2.0	822.2	8.035
16,384	1/2	4,164,292.5	2911.3	3.6	12,590.0	2682.876
	1/4	2,885,712.1	1356.4	1.9	3748.1	338.651
	1/8	2,192,246.5	549.4	1.6	830.3	26.047
32,768	1/2	8,335,031.6	3442.4	2.6	12,449.2	13,600.522
	1/4	5,765,504.1	1703.5	1.8	3911.1	2095.001
	1/8	4,389,332.8	926.7	1.1	1538.7	235.187

(*continued on next page*)

Table 1 (*continued*)

n	λ	$z^{\star}$	♯DC	♯KSP	♯KP	CPU (seconds)
Panel C: UNCOR, s = 8						
512	1/2	124,412.4	175.2	4.5	1693.8	0.158
	1/4	76,073.3	74.9	2.6	746.7	0.051
	1/8	49,696.0	26.4	2.1	291.0	0.020
1024	1/2	247,913.5	452.2	5.3	4100.8	1.306
	1/4	152,083.2	155.9	3.2	1428.1	0.279
	1/8	99,697.9	49.5	1.8	449.6	0.083
2048	1/2	500,175.7	821.2	6.7	7361.1	8.334
	1/4	303,629.9	316.2	3.4	2670.2	1.689
	1/8	199,779.0	82.4	2.0	628.1	0.324
4096	1/2	1,003,363.9	1407.4	6.2	12,375.6	56.343
	1/4	610,181.6	467.2	3.2	3740.4	8.146
	1/8	398,089.8	181.1	1.7	1308.7	1.992
8192	1/2	2,007,457.8	2270.9	6.6	20,148.0	444.922
	1/4	1,221,637.3	685.4	2.3	5294.0	45.467
	1/8	798,697.7	259.4	1.6	1579.8	7.792
16,384	1/2	4,006,383.7	2933.1	5.6	25,640.2	3130.824
	1/4	2,445,181.6	1018.0	1.7	7728.6	337.385
	1/8	1,598,601.3	434.7	1.5	2399.4	40.899
32,768	1/2	8,020,818.2	3640.1	4.7	31,579.0	18,134.677
	1/4	4,881,680.3	1223.9	1.8	8325.3	2045.644
	1/8	3,200,604.0	611.9	1.5	2553.9	225.649

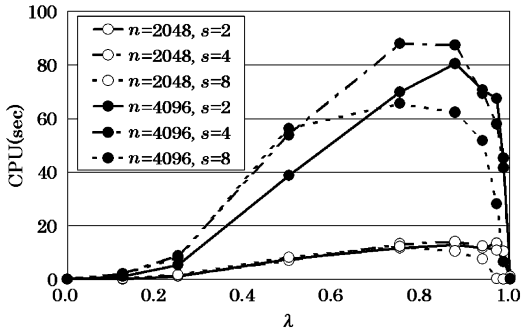

Fig. 5. CPU seconds vs. λ.

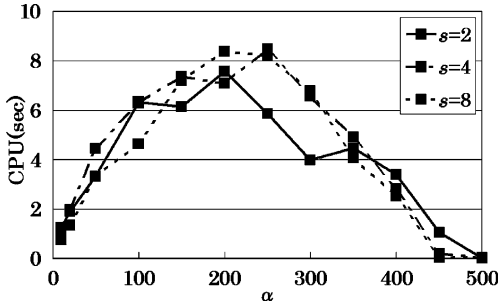

Fig. 6. CPU seconds vs. knapsack capacity ($n = 2048, \lambda = 1/2$).

Table 2
The result of experiment

n	λ	$z^{\star}$	♯DC	♯KSP	♯KP	CPU (seconds)
Panel A: WEAK, s = 2						
512	1/2	127,519.4	238.4	3.3	565.2	0.550
	1/4	105,164.2	4.4	1.0	18.4	0.012
	1/8	87,180.9	3.3	1.0	16.5	0.018
1024	1/2	254,911.4	422.3	2.7	1032.9	2.771
	1/4	211,701.8	4.1	1.0	16.9	0.032
	1/8	175,337.4	3.4	1.0	15.0	0.030
2048	1/2	511,051.1	601.5	3.4	1425.7	9.665
	1/4	424,127.6	3.7	1.0	14.9	0.065
	1/8	352,725.4	3.4	1.0	15.6	0.074
4096	1/2	1,022,660.1	769.2	2.8	1343.0	32.135
	1/4	846,463.8	3.6	1.1	13.9	0.172
	1/8	704,942.3	3.1	1.0	13.4	0.133
8192	1/2	2,045,190.8	1041.2	2.9	1502.0	171.262
	1/4	1,690,568.5	3.5	1.1	14.3	0.600
	1/8	1,408,351.9	3.4	1.3	15.1	0.465
16,384	1/2	4,090,299.4	1364.2	3.6	1731.9	969.199
	1/4	3,382,300.0	3.8	1.3	13.0	3.006
	1/8	2,813,502.7	3.4	1.6	12.6	1.647
32,768	1/2	8,181,635.5	1762.6	6.3	1957.0	4853.373
	1/4	6,762,091.4	4.3	1.6	12.9	15.181
	1/8	5,629,033.8	3.8	1.5	10.3	5.738
Panel B: WEAK, s = 4						
512	1/2	126,892.1	101.6	3.4	484.6	0.225
	1/4	90,496.8	3.9	1.0	32.0	0.010
	1/8	62,224.7	3.3	1.0	29.3	0.012
1024	1/2	253,609.3	199.3	3.5	950.0	1.272
	1/4	183,184.0	3.7	1.0	29.2	0.022
	1/8	125,590.6	3.5	1.0	29.6	0.030
2048	1/2	508,314.7	306.2	3.2	1421.2	4.629
	1/4	366,548.6	3.5	1.0	27.5	0.063
	1/8	253,698.6	3.3	1.0	26.7	0.070
4096	1/2	1,017,781.5	406.2	4.0	1793.8	17.440
	1/4	730,029.7	3.6	1.0	25.1	0.164
	1/8	506,959.7	3.3	1.0	24.7	0.140
8192	1/2	2,035,157.5	567.9	4.2	2328.1	98.996
	1/4	1,456,232.2	3.5	1.0	23.1	0.464
	1/8	1,010,978.5	3.2	1.0	22.0	0.284
16,384	1/2	4,070,231.1	722.2	3.8	2542.9	554.726
	1/4	2,914,961.1	3.3	1.1	20.3	1.833
	1/8	2,017,694.9	3.2	1.2	21.0	0.848
32,768	1/2	8,141,805.1	914.3	3.2	2115.3	2706.472
	1/4	5,826,061.8	3.7	1.3	19.7	9.442
	1/8	4,038,326.1	3.3	1.1	16.8	3.324

(*continued on next page*)

Table 2 (*continued*)

n	λ	$z^\star$	♯DC	♯KSP	♯KP	CPU (seconds)
Panel C: WEAK, s = 8						
512	1/2	126,847.0	8.4	1.0	9.6	0.016
	1/4	83,150.0	4.6	1.0	69.1	0.011
	1/8	49,745.1	3.6	1.0	59.0	0.012
1024	1/2	253,418.2	68.3	1.9	452.3	0.410
	1/4	168,917.5	4.2	1.0	57.4	0.024
	1/8	100,704.8	3.6	1.0	56.8	0.028
2048	1/2	507,845.8	210.8	4.1	1812.6	3.110
	1/4	337,759.4	3.8	1.0	55.6	0.059
	1/8	204,181.8	3.7	1.0	54.5	0.067
4096	1/2	1,016,903.4	287.0	4.3	2451.6	11.421
	1/4	671,807.3	3.7	1.0	46.9	0.146
	1/8	407,966.9	3.3	1.0	44.9	0.125
8192	1/2	2,033,541.4	380.1	4.6	3084.4	65.622
	1/4	1,339,059.3	3.7	1.0	44.7	0.458
	1/8	812,289.1	3.4	1.0	42.4	0.271
16,384	1/2	4,067,029.5	533.5	4.8	4191.4	397.001
	1/4	2,681,285.0	3.7	1.0	40.2	1.909
	1/8	1,619,794.0	3.2	1.0	38.5	0.688
32,768	1/2	8,135,309.6	615.8	4.9	4130.4	1784.810
	1/4	5,358,043.8	3.3	1.0	34.3	6.643
	1/8	3,242,973.2	3.2	1.1	32.7	2.268

Table 3
The result of experiments (STRONG)

n	s	λ	$z^\star$	♯DC	♯KSP	♯KP	CPU (seconds)
256	2	1/4	52,026.7	9.2	1.0	20.9	0.454
		1/8	42,683.6	5.7	1.0	18.8	13.124
	4	1/4	44,870.9	7.7	1.0	36.3	0.037
		1/8	30,477.1	4.8	1.0	31.6	0.443
	8	1/4	41,298.8	6.8	1.0	61.0	0.012
		1/8	24,381.8	4.6	1.0	58.3	0.011
512	2	1/4	104,073.9	15.2	1.0	28.3	169.739
		1/8	85,804.4	9.6	1.0	21.3	3905.249
	4	1/4	89,928.3	10.1	1.0	33.8	0.777
		1/8	61,772.7	6.4	1.0	30.2	51.819
	8	1/4	82,856.6	8.0	1.0	59.1	0.104
		1/8	49,753.6	5.9	1.0	58.0	0.393
1024	8	1/4	166,363.1	13.5	1.0	64.7	3.275
		1/8	99,783.3	8.3	1.0	55.6	42.066

- Strongly correlated case (STRONG)
  w_j: Uniformly and independently random over $[1, 1000]$.
  p_j: $p_j = w_j + 100$.

Table 2 Panels (A)–(C) and Table 3 summarize the results of the WEAK and STRONG cases respectively. Comparing Tables 1 and 2, we observe the following:

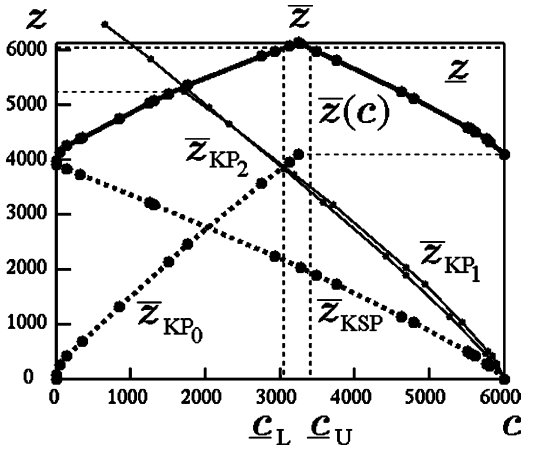

Fig. 7. Function $\bar{z}(c)$ for the case of WEAK.

1. For the same values of n, λ and s, ♯DC and ♯KP are substantially smaller in WEAK than in UNCOR.
2. Consequently, CPU time is also smaller in WEAK.

For the problem with $n = 2^{15}$ items weakly correlated instances can be solved 10–100 times faster than uncorrelated counterparts.

The reason for this is explained in Fig. 7 (For the details, refer to [20]). Correlation between weights and profits makes functions $\bar{z}_{\mathrm{KP}}(c)$ and $\bar{z}_{\mathrm{KSP}}(c)$ almost a straight line, and thus their sum $\bar{z}(c)$ is pointed at $(\bar{c}, \bar{z})$. Then, the interval between $\underline{c}_L$ and $\underline{c}_U$ is much smaller in correlated case than in uncorrelated case, and thus in UNCOR we need to examine a wider interval; consequently the CPU time to do this is longer.

However, in the strongly correlated case, the problem is hard to solve. Indeed, in this case we were only able to solve problems with $n \leqslant 1024$, as shown in Table 3. This is because the Horowitz–Sahni method [7] used to solve KPs in our algorithm is very inefficient for such type of problems.

7. Conclusion

In this paper we have formulated the GKSP, developed a solution algorithm, and conducted some numerical experiments. Our algorithm outperformed a commercial software, and we were able to solve problems with up to $n = 2^{15}$ items to optimality. Strongly correlated case remains difficult to solve, and this is left for future work.

Appendix A. Finding all the discontinuity points of $z^{\star}_{\mathrm{KP}}(c)$

In the algorithms of Section 4, we had to list up all the discontinuity points of $z^{\star}_{\mathrm{KP}}(c)$ within $[\underline{c}_L, \underline{c}_U]$. We may use the dynamic programming approach of Nemhauser and Ullman [13] for this purpose. However, this method is primarily for the interval of the form $[0, c]$, and is not efficient enough for $[\underline{c}_L, \underline{c}_U]$, especially when we have large $\underline{c}_L$ and relatively small $\underline{c}_U - \underline{c}_L$.

Here we give an algorithm to list up all the discontinuity points of $z^{\star}_{\mathrm{KP}}(\cdot)$ within the interval $[\underline{c}_L, \underline{c}_U]$. First, we introduce the following *inverse knapsack problem*.

$\mathbf{IKP}_k(z)$:

$$\text{Minimize} \quad \sum_{j\in N_k} w_j x_j \tag{31}$$

$$\text{subject to} \quad \sum_{j\in N_k} p_j x_j \geqslant z, \tag{32}$$

$$x_j \in \{0,1\}, \quad j \in N_k. \tag{33}$$

If (33) is continuously relaxed to $0 \leqslant x_j \leqslant 1$, we have problem $\underline{\mathrm{IKP}}_k(z)$. Then, the following holds.

Theorem 4. *The optimal objective values of* $\mathrm{IKP}_k(z)$ *and* $\underline{\mathrm{IKP}}_k(z)$ *are, respectively,* $c^{\star}_{\mathrm{KP}_k}(z)$ *and* $\underline{c}_{\mathrm{KP}_k}(z)$.

Proof. These are obvious from the definitions of $c^{\star}_{\mathrm{KP}_k}(z)$ and $\underline{c}_{\mathrm{KP}_k}(z)$. □

In what follows, we are concerned with $\mathrm{IKP}_0(z)$, and for simplicity suffix 0 is dropped unless otherwise stated. The following algorithm solves our problem:

Algorithm LISTUP_DC_POINTS(Generic)

Step 1. Set $c := \underline{c}_U$.
Step 2. If $c \leqslant \underline{c}_L$ stop. Otherwise solve $\mathrm{KP}(c)$ exactly to obtain $z^{\star} := z^{\star}_{\mathrm{KP}}(c)$.
Step 3. Solve $\mathrm{IKP}(z^{\star})$ to obtain the optimal objective value $c^{\star} := c^{\star}_{\mathrm{KP}}(z^{\star})$. Output $(c^{\star}, z^{\star})$ as a discontinuity point.
Step 4. Set $c := c^{\star} - 1$ and go back to **Step 2**.

Here we note that $\mathrm{IKP}(z)$ can be solved by converting it into a standard knapsack problem through the change of variables $y_j := 1 - x_j$ $(j \in N)$. Thus, in the above algorithm we need to solve two knapsack problems per discontinuity point. Using the notation of Section 5.4, we have

$$\#\mathrm{KP} = 2\#\mathrm{DC}. \tag{34}$$

We may further reduce the number of KPs we need to solve to list up all the discontinuity points in the following way. That is, in Step 2 if we use Horowitz–Sahni algorithm (H–S method for short, [7]) to solve $\mathrm{KP}(c)$ and obtain an optimal solution $\boldsymbol{x}^{\dagger}$ with the objective value $z^{\dagger} := \sum_{j\in N_0} p_j x^{\dagger}_j$ and weight $c^{\dagger} := \sum_{j\in N_0} w_j x^{\dagger}_j$, $(c^{\dagger}, z^{\dagger})$ frequently happens to be a discontinuity point. Then, the algorithm is revised as follows.

Algorithm LISTUP_DC_POINTS

Step 1. Solve $\mathrm{KP}(\underline{c}_U)$ using H–S method to obtain $(c^{\dagger}, z^{\dagger})$.
Step 2. If $c^{\dagger} \leqslant \underline{c}_L$ stop. Otherwise solve $\mathrm{KP}(c^{\dagger} - 1)$ by H–S method, and obtain (c', z').
Step 3. If $z' = z^{\dagger}$, let $c^{\dagger} := c'$ and go to **Step 2**.
Step 4. Output $(c^{\dagger}, z^{\dagger})$ as a discontinuity point. Let $(c^{\dagger}, z^{\dagger}) \leftarrow (c', z')$, and go back to **Step 2**.

If in the above algorithm Step 3 never occurs, we have #KP = #DC. In practice, in the numerical experiments of Sections 5 and 6, this was usually

$$\#\mathrm{KP} \leqslant 1.05\#\mathrm{DC}. \tag{35}$$

Comparing this against (34), we see that the revised algorithm is approximately twice faster than the original.

References

[1] J.R. Brown, The knapsack sharing problem, Operations Research 27 (1979) 341–355.
[2] J.R. Brown, Bounded knapsack sharing, Mathematical Programming 67 (1994) 343–382.
[3] D.-Z. Du, P.M. Pardalos (Eds.), Minimax and Applications, Kluwer, 1995.
[4] R. Fourer, Software survey: Linear programming, OR/MS Today 26 (1999) 64–71.
[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman and Company, 1979.
[6] M. Hifi, S. Sadfi, The knapsack sharing problem: An exact algorithm, Journal of Combinatorial Optimization 6 (2002) 35–54.
[7] E. Horowitz, S. Sahni, Computing partitions with applications to the knapsack problem, Journal of ACM 21 (1974) 277–292.
[8] S. Kaplan, Application of programs with maximin objective functions to problems of optimal resource allocation, Operations Research 22 (1974) 802–807.
[9] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer-Verlag, 2004.
[10] T. Kuno, H. Konno, E. Zemel, A linear-time algorithm for solving continuous maximin knapsack problems, Operations Research Letters 10 (1991) 23–26.
[11] H. Luss, Minimax resource allocation problems: Optimization and parametric analysis, European Journal of Operational Research 60 (1992) 76–86.
[12] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, 1990.
[13] G.L. Nemhauser, Z. Ullmann, Discrete dynamic programming and capital allocation, Management Science 15 (1969) 494–505.
[14] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, John Wiley & Sons, 1988.
[15] Mathematical Systems Inc., NUOPT Manual, Available from: <http://www.msi.co.jp/nuopt>, 2002.
[16] C.S. Tang, A max-min allocation problem: Its solutions and applications, Operations Research 36 (1988) 359–367.
[17] T. Yamada, M. Futakawa, Heuristic and reduction algorithms for the knapsack sharing problem, Computers & Operations Research 24 (1996) 961–967.
[18] T. Yamada, M. Futakawa, S. Kataoka, Some exact algorithms for the knapsack sharing problem, European Journal of Operational Research 106 (1998) 177–183.
[19] T. Yamada, H. Takahashi, S. Kataoka, A branch-and-bound algorithm for the mini-max spanning forest problem, European Journal of Operational Research 101 (1997) 93–103.
[20] T. Yamada, Available from: <http://www.nda.ac.jp/~yamada/ypublication.html>, 2004.